G

Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks

L.T. Heberlein, B. Mukherjee, K.N. Levitt

Computer Security Laboratory
Division of Computer Science
University of California
Davis, Ca. 95616

ABSTRACT

The model for an Internetwork Security Monitor (ISM) is presented. The objective of the model is to significantly improve our capability to detect and react to intrusions into an arbitrary wide-area network (WAN) (e.g., the Internet) through a distributed intrusion-detection and analysis system. The system will monitor the various component networks of the internetwork and bring potentially intrusive behavior to the attention of the local-network security managers. The model primarily extends the DIDS and NSM intrusion-detection systems and takes advantage of, but does not require, cooperative host monitoring. This design will provide the first intrusion-detection system that aggregates information from different monitors over wide-area networks and will be deployable at different sites with widely different operating environments and security requirements.

1. INTRODUCTION

The prevalence and ease of use of networking to provide remote access to resources has brought with it a set of previously unanticipated problems. Network managers worldwide are extremely concerned with the problem of network intrusions, which are unwanted or unauthorized use of the network to gain access to (and sometimes modify) both network and computing resources. These intrusions have often appeared in the popular news media and have been a serious impediment to many organizations obtaining network connection.

What do we mean by network intrusion? For our purposes here, we consider a network intrusion to be any unwanted or unauthorized actions being taken across the network that affect remote resources. These actions include those of the "Wily Hacker" [Sto89]—where the intruder aims to gain unauthorized access to information on a number of computers on the network—unauthorized remote modifications of router tables in an Internet, and attempts to deny use of the network to authorized users.

Examples of network intrusions that concern operators include:

- unauthorized modifications of system files that permit unauthorized access to either system or user information
- unauthorized access to user file space
- unauthorized modifications of user files/information
- unauthorized modifications of tables or other system information in network components
- unauthorized use of computing resources (perhaps through the creation of unauthorized accounts or through the unauthorized use of existing accounts.)

Intrusion Detection Systems (IDS) attempt to detect the presence of such attacks. Early IDS were designed around the analysis of a single host's audit trail. Their examples are SRI's early model of the Intrusion Detection Expert System (IDES) [Den87], National Security

Agency's MIDAS, Haystack Laboratories' Haystack System [Sma88], Los Alamos National Laboratory's Wisdom & Sense (W&S) [Vac89], and AT&T's ComputerWatch [Dow90]. However, with the proliferation of computer networks, many of these IDS began to apply their host based techniques to small networks of computers. Their examples include SRI's IDES [Lun90] and the Distributed Intrusion Detection System (DIDS) [Sna91].

Unfortunately, even with the extension of IDS into small networks of computers, because the networks are often interconnected, an IDS's ability to detect intrusive activity and to determine the party responsible for such activity is limited. Intrusion detection is an inter-network problem. Often intrusions or attacks affect more than a single network, and detection may require exploitation of data from multiple networks or computers.

To address these limitations, we designed a model, called the Internetwork Security Monitor (ISM), to perform intrusion detection in a highly interconnected wide-area network. Specifically, our ISM design requires the development of a hierarchical internetwork monitor as an extension of ongoing work in distributed-intrusion detection. In extending the LAN monitoring capabilities into an internetwork environment, we are exploring the feasibility of different design alternatives for distributed-network traffic monitoring and analysis, including the following hierarchical architecture. Under this architecture, independent monitors are placed at various locations over an internetworked environment. These monitors exchange and share information (including those on hypothesized attacks) to detect possible security breaches. Subnetworks, in turn, exchange information among one another to detect inter-subnetwork attacks.

The scenarios in Section 2 motivate our work by describing the type of behavior that our internetworked security monitor model is designed to detect and analyze. Section 3 presents an overview of the ISM components. Section 4 discusses how complete accountability can be attained in a networked environment. Section 5 presents the ISM model as an extension of current work being done. Finally, Section 6 provides some concluding remarks.

2. SCENARIOS

Because the Internet is distributed, the evidence needed to detect an intrusion may also be distributed across the Internet. For example, suppose an intruder systematically attacks hosts at a particular organization (site A) until he successfully penetrates a host. This attack method, called the doorknob attack, may be successfully detected at site A. However, once the intruder has acquired a foothold on a computer at site A, he may notice a *.rhosts* file in a user's home directory, which indicates that the user trusts logins from computers at a second organization (site B). Hoping the trust is mutual (i.e., the user has *.rhosts* files in his accounts at site B for logins from site A), the intruder could masquerade as this user at site A and successfully log into a computer at site B.

Since this login would be between two machines which do occasionally exchange logins, and since no vulnerabilities other than trust would be exploited, site B's intrusion-detection system would be unable to discern this login as an intrusion.

A second scenario is based on an actual attack detected and analyzed by the Network Security Monitor (NSM) [Heb91]. This attack also begins as a doorknob attack. The intruder, attacking across the Internet from site A, attempts to penetrate over sixty computers before eventually finding one with the default (and flawed) system configuration in place. Once the intruder penetrates this host, the attacker quickly inserts a Trojan login program, and prior to the intruder exiting the penetrated machine, a login from site B successfully exploits the newly installed Trojan login program. The intruder from site B remains logged in for several hours exploiting various bugs in systems as well as the trust between the organization's machines.

Investigation the next day shows that neither of the hosts from sites A or B were the root of the attacks. Their systems had also been attacked and merely used as launch pads to attack the computers.

The next night, the intruder penetrates the machines from a third organization (site C). Detecting the penetration, we examined the host at site C as the intruder, but the host at site C, obviously subverted, reports that no one is logged on. Our trail has gone cold again.

From these and other incidents, we are convinced that we have very little chance of catching intruders originating outside our organization. With current intrusion-detection techniques, we can detect many intrusions into our systems, but attacks from outside are relatively difficult to dissect.

3. ARCHITECTURE OVERVIEW

The ISM model extends research and development efforts already existing in the field of intrusion detection. Primarily, the ISM extends the Distributed Intrusion Detection System (DIDS) (see [Sna91]) into arbitrarily wide networks. Multiple DIDS-like monitors, called ISM domain monitors, communicating through well-defined protocols form the core of the distributed ISM. In addition to the monitors themselves, Security Domain Name Servers (SDNS), based on the Domain Name Server (DNS) model, provide a mechanism for the ISMs to locate each other across the Internet. Finally, security workbenches allow network managers to logon to their local ISM domain monitor to examine the results of the monitor's analysis, query further into possible intrusions, exchange information with other network security managers, and administer various security tools such as Security Profile Inspector (SPI) or Computer Oracle Password Security system (COPS). Although all three major components—ISM domain monitors, Security Domain Name Servers, and security workbenches—comprise the ISM model, this paper focuses on the ISM domain monitors.

4. ACCOUNTABILITY

One of the most fundamental and critical capabilities in a computer system security is establishing accountability for actions performed by individuals. A combination of authentication and auditing mechanisms residing in the operating system usually provides this accountability. In such systems, the user identifies and verifies himself (via a password) to the authentication mechanism, and the auditing mechanism keeps account of the activities performed by that authenticated user.

Unfortunately, the accountability can be lost when the user crosses operating system boundaries (e.g., logging into another host across the network). Although the user will be re-authenticated by the new machine (either with a new password or by trusting the authentication of the first host), the accounting of the user's activities will be distributed across the audit trails of multiple hosts. If users are restricted from changing their identification as they move across systems, and if the audit timing can be synchronized across auditing mechanisms, accountability can be achieved; however, such restrictiveness is not attainable in many environments.

4.1 NETWORK IDENTIFIER

The Distributed Intrusion Detection System (DIDS) was designed in part to achieve an accountability across a network of heterogeneous systems. When a user initially signs on to one of the components of the network, that user is assigned a Network Identifier (NID). As the user moves across the network of computers, all activity performed by that user on any host is mapped to the NID. Therefore, accountability across the network is established.

To account for a user's activities across the network, DIDS, working with the established auditing mechanism on each host, creates a map between a user's UID for a session and an NID. A user's activities across a network can be accounted for by extracting the activities from each host associated with a UID which maps to the same NID.

DIDS creates the map between the UID on a host and the NID by tracking a user's movement across the network and exploiting transitivity. For example, if user A on host1 performs a remote login across the network to host2 as user B, DIDS tracks A@host1 to B@host2, and the instance B@host2 is mapped to the same NID as A@host1. If the user performs a second remote login from host2 to host3 as user C, we can use the simple rules

NID(C@host3) = NID(B@host2)
and NID(B@host2) = NID(A@host1)

to conclude that the NID for C@host3 is the same as the NID for A@host1.

The tracking between users and hosts is performed by treating a network connection as a shared resource and determining which users are accessing that resource. For example, if a user creates a remote login session (called session2), on host2, DIDS first identifies the host-to-host connection (called net-rsrc) responsible for the session and binds the information as the pair <net-rsrc, session2@host2>. DIDS then determines which session on host1 meets the requirements <net-rsrc, ?@host1>, and tracking is achieved (see Figure 1).

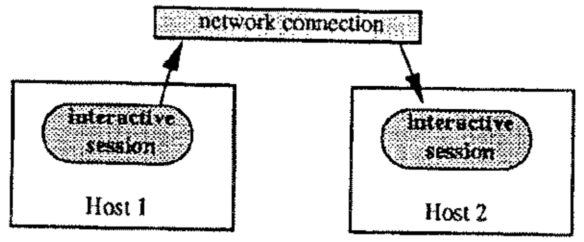

Figure 1. Network Connection as a Shared Resource

Unfortunately, in many environments, not all computers on the network support a host monitor which provides either the accountability for that particular host or the information required to track users across the network. In such environments, security and accountability can be increased by using a network monitor such as the Network Security Monitor (NSM).

4.2 TRACKING USERS THROUGH HOSTS WITHOUT MONITORS

The NSM, initially designed to detect intrusive activity across a local-area network (LAN), already augments DIDS' analysis capability by scrutinizing network activity into hosts which do not support host monitors; therefore, all hosts in the DIDS domain can be monitored to a certain level for the presence of intrusive activity. However, the current DIDS system cannot perform the NID tracking when a user passes through an unmonitored host. The following example illustrates the problem.

Suppose the DIDS-monitored domain consists of three hosts, two of which support host monitors (hosts one and three) and one which does not (host two). In this domain, a person initially signing onto host1 and then performing a remote login to host3 will have all of his activities mapped to a single NID, so DIDS maintains complete accountability. However, if the person first performs a remote login to host2 (the unmonitored host) and then performs a second login from host2 to host3, the user's activities on host3 will not be mapped to the same NID as his activities on host1, so DIDS loses complete accountability (see Figure 2).

Our challenge has, therefore, expanded to obtaining complete accountability across all monitored hosts by mapping a user's activities on all these hosts to the same NID even if the user temporarily leaves the domain of monitored hosts. Fortunately, the mapping of a user's activities can still be obtained by tracking the user by connections, even through unmonitored hosts, if we simply expand our notion of a network connection.

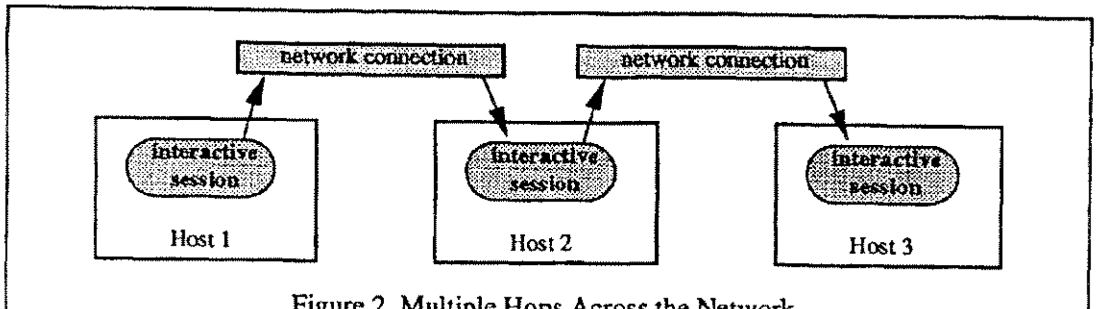

Figure 2. Multiple Hops Across the Network

4.2.1 EXTENDED CONNECTIONS

The previously described DIDS algorithm attains a map of a UID to an NID by tracking the user's login session back to the original login session. DIDS performs its tracking by determining the ownerships of a shared resource, namely a network connection, and recursively applies the procedure until DIDS reaches the original login session. However, the recursion fails when one of the hosts involved is an unmonitored host. The following conceptual extension to network connections allows us to continue the recursive algorithm through unmonitored hosts.

In the previous example, a user on host1 performs a remote login to host2 and then performs a second remote login to host3. Figure 2 presents a logical view of the user's actions. Using some I/O device (e.g., a terminal) connected to the session on host1, the individual can perform actions on host3 and view the results as if he were connected directly to the session on host3. This "virtual," direct connection occurs because the session on host2 is acting as a repeater. Thus, all commands and results are passed through the session on host2 unaltered. By exploiting this invariance, we can view the two network connections in Figure 2 as components of a single "extended" connection between the session on host1 and the session on host3. Now when the DIDS recursive algorithm used to track users encounters an unmonitored host, the algorithm can bypass the host by exploiting the extended connection, if one exists, and continue the algorithm at the next monitored host.

Formally, we define an extended connection as a set of network connections used to transport data and control between two sessions. Figure 3 shows an extended connection in relationship to data, host-to-host protocols, point-to-point protocols, intermediate sessions, and routers. As Figure 3 shows, only the data (e.g., control information sent from host1 to host3) remains invariant across the various network components. By exploiting this invariance, via a method we call thumbprinting, the NSM maps the various host-to-host connections to the same extended connection.

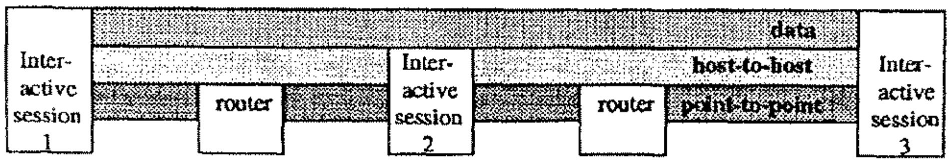

Figure 3. Extended Network Connection

4.2.2 THUMBPRINTS

The NSM maps host-to-host connections to an extended connection by assigning to each host-to-host connection a thumbprint representing the data flow for that connection for a specified period of time and then comparing the thumbprints for the various connections. If the thumbprints for two host-to-host connections match (within a measure of tolerance), they are mapped to the same extended connection. Thumbprinting even works when the number of intermediate, unmonitored hosts, between two host-to-host connections is unknown (see Figure 4).

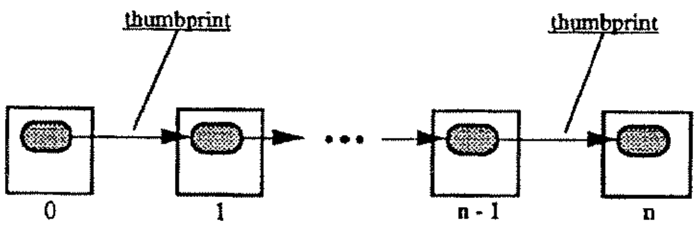

Figure 4. Tracking a User Across Unmonitored Hosts

We formally define a thumbprint for a host-to-host connection as a vector, $\mathbf{X} = \langle x_1, x_2, \ldots, x_n \rangle$, where each x_i is a counter for the occurrence of some attribute in the data. Two thumbprints, **X** and **Y**, are compared for similarity by determining the distance between the two vectors, $|\mathbf{X} - \mathbf{Y}|$. The certainty that the two connections which created the thumbprints are actually part of the same extended connection is inversely related to this magnitude.

The mapping of data in a host-to-host connection to a thumbprint vector, although extremely important, is not necessarily uniquely defined, and we are experimenting with various implementations. The implementations are driven by several goals described in Table 1.

Table 1. Thumbprint Implementation Goals

Resolution	The primary purpose of the thumbprint is to correctly recognize that two host-to-host connections are part of the same extended connection.
Semantic Free	As will be seen later, thumbprinting will be used in an open environment where privacy is an issue; therefore, the thumbprint, while being able to represent the connection, should not reveal the contents of the connection data.
Efficiency	The calculation of each x_i as well as the calculation of $|\mathbf{X} - \mathbf{Y}|$ must be efficient in order to allow for the thumbprinting of thousands of simultaneous connections and their comparison in real time.

Up to this point, we have argued for the need for accountability in computer systems, and we have shown that for complete accountability across operating system boundaries, we need to be able to track users across the boundaries. DIDS has proven that tracking can be performed in a small network of monitored hosts, and we have described an extension to DIDS allowing us to track users across an unknown number of unmonitored hosts. We now present an architecture based on NSM and DIDS which provides for intrusion detection and accountability in large-scale interconnected networks (e.g., the Internet).

5 ISM

The ISM model links together security systems monitoring particular domains (e.g., a DIDS-monitored domain) via standard information exchange protocols such as the Common Management Information Protocol (CMIP) to create a large-scale, highly distributed intrusion-detection system. The model is flexible in that different security domains can choose their own level of security analysis, from virtually none to complete transaction-to-transaction analysis, as long as they provide a minimum set of functionality described below. Finally, the model is hierarchical and supports the current network management structure by hiding a site's internal security structure from outsiders.

5.1 ISM PEER-LEVEL COMMUNICATION

An ISM is responsible for a specific set of hosts. When a user initiates a connection from a host in one ISM domain to a host in a second ISM domain, the ISMs may exchange information to allow a more accurate analysis of the security state of their own domains. At a minimum, an ISM must be able to identify the source (local or external to the domain) for connections leaving its domain. If the user initiating the connection originated inside the ISM domain, the ISM need only respond that the connection began internally and not reveal the actual origin of the user. If the connection originated outside the ISM domain (e.g., the user merely passed through the domain), the ISM must respond with the host-to-host connection definition of the connection entering the domain. This minimum capability of an ISM prevents an intruder from exploiting the domain in an attempt to disguise his origin. The protocol to support this functionality is presented below:

- `GET TIME <time>`
- `GET CONNECTION TCP/IP-DEF <def> TIME <time>`
- `GET ORIGIN CONN-ID <id>`

The first request allows an ISM to synchronize its clock to the remote ISM. An alternate, and preferred method is to assume all monitors are running under a time protocol (e.g., the network time protocol, NTP). The second request (with the time given in the remote ISM's time frame) returns an identifier, which can be used to make further requests. The third request, fulfilling the minimum requirement for an ISM, returns the origin of the user (relative to the local ISM) as either local to the domain or external (including the TCP/IP-DEF).

Other functionality for an ISM, while helpful but not required, includes the ability to analyze the activity within the domain for intrusive activity. Access to this analysis by external ISMs are made by the following requests:

- `GET ANALYSIS CONN-ID <id>`
- `GET ANALYSIS HOST-ID <host-address>`
- `GET ANALYSIS SERVICE <service-name>`
- `GET ANALYSIS VULNERABILITY <vulnerability-id>`

The first request returns a value between 0 and 100, which indicates whether or not the ISM believes that the user owning the connection given by <id> is behaving intrusively. The second request also returns a value between 0 and 100, indicating whether or not the ISM believes that the host is associated with intrusive activity. The host does not necessarily have to be within the ISM's domain. For example, if one ISM believes it is receiving a number of possibly intrusive connections from a particular host, it can query other ISMs as to whether they believe the host has a hostile user on it. The third request returns a value between 0 and 100 indicating the ISM's belief that service <service-name> is being used in an unusual and intrusive manner (e.g., when the Internet worm exploited a hole in the mail service). The last request returns a value between 0 and 100 indicating the ISM's belief that a particular vulnerability has recently been exploited. To perform this, the ISM must have a catalog of known vulnerabilities and signatures to detect their [attempted] exploitation. Due to the sensitive nature of vulnerabilities, some ISMs (e.g., those at government sites) may have a more complete listing than other ISMs (e.g., those at universities).

As an example, Figure 5 shows three ISM domains in which a single user accesses hosts in all three domains. ISM1 is able to observe all hosts within its domain; however, the hosts inside the second and third domains are hidden from ISM1's view. When a user connects to ISM1's domain, ISM1 queries ISM2 for the source of the connection, and ISM2 responds that the source is external and supplies the TCP/IP definition of the connection to ISM1. ISM1 can use this definition to query ISM3 and determine whether the source of the connection into ISM1 is somewhere inside of ISM3.

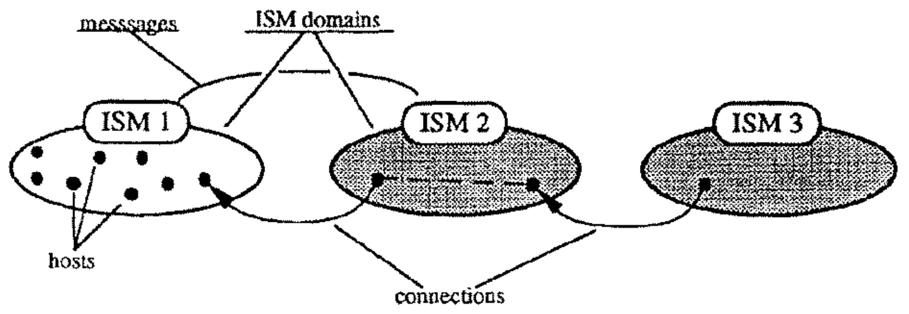

Figure 5. ISM Security Domains

5.2 ISM HIERARCHICAL COMMUNICATIONS

The ISM model also allows ISMs to be grouped hierarchically. For example, ISM1' may monitor a domain which is divided into three sub-domains, each with its own ISM sub-monitors. This hierarchical structure provides two major benefits. First, because the ISM1' domain can look into its sub-domains, it can aggregate a user's activities across these sub-domains. This functionality is provided by additional requests which can only be made by a direct parent ISM. These requests, however, can only be answered if the ISM sub-domain monitors support full tracking and accountability. The protocol to support these request are:

- `GET NID CONN-ID <id>`
- `GET PATH NID <nid>`
- `GET VECTOR NID <nid>`

The first request returns the NID associated with a given connection ID, and the NID can be used as a key to request further information. The second request returns an NID trace showing a user's movement throughout the domain. The third request returns a NID vector—a list of counts representing a user's activities in different categories (e.g., number of files opened or the number of times a specific command has been executed)—for the user in that sub-domain. If a user's activities crosses several sub-domains, the parent ISM can trace all of the user's activities by requesting the paths and vectors of the user across all the sub-domains he crosses.

The second benefit of the ISM's hierarchical architecture is that internal structure can be hidden from outsiders. An individual site (e.g., a university or government research facility) may contain only a single ISM monitor (e.g., monitoring all traffic in and out of the site), or it may contain many sub-domains, each with its own ISM, divided along department lines. Whatever the structure, external sites can only view the site as a single ISM. The following example illustrates this ISM encapsulation, or information hiding.

Site A is composed of three ISM sub-domains, and site B is composed of two sub-domains. When a host in site A's third domain connects to a host in site B's first domain, site B's first domain cannot "see" site A's domain hierarchy, so it must send all queries to site A's parent ISM. Likewise, if site A's third domain queries site B for an analysis of the connection, the domain must send the query to site B's parent ISM. Importantly, to protect site A's internal structure, site A's third domain monitor performs its query through site A's parent ISM. Otherwise, a user at site B could determine site A's internal structure by "probing" site A and observing which internal ISMs respond to which probes. Meanwhile, site A's internal ISM domain monitors may continue to query each other locally (see Figure 6).

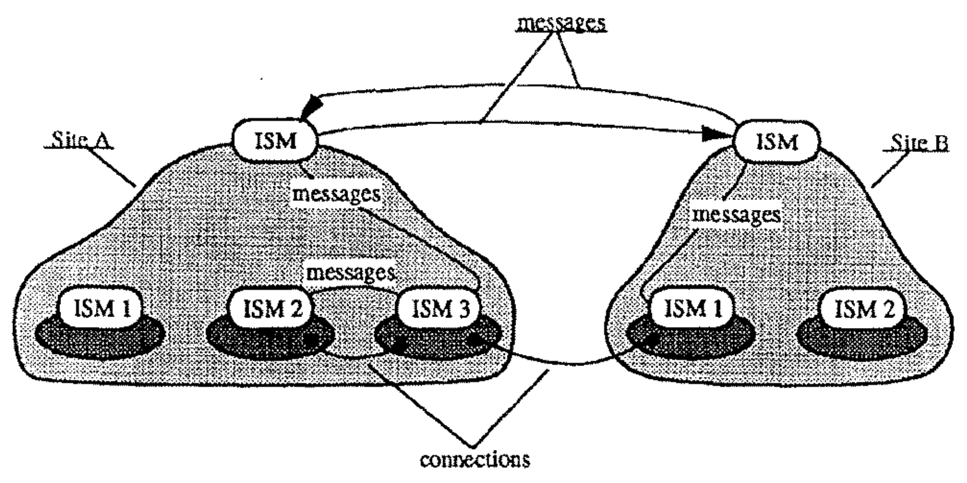

Figure 6. Security Domain Hierarchy

6. CONCLUSIONS AND FUTURE RESEARCH

Wide-area networks (e.g., the Internet) have grown to be large and complex, consisting of several thousands of networks (both wide-area and local-area) and managed by a comparably large number of organizations. Providing for coordinated network management in such an environment is a major task—one requiring advanced technologies that utilize the network and computing tools to assist in the management process. Nowhere does this issue show up more than in the area of network security. Because the Internet is distributed, evidence to identify and analyze an intrusion can be distributed over multiple sites on the Internet. Network managers at each site on the Internet must be provided with tools to analyze the evidence of an intrusion at the site and with tools to communicate their evidence and analysis with other managers so that the intrusion can be understood. The proposed ISM design focuses on providing a distributed, intelligent, decision-support system for network managers that would partially automate the detection of intrusions into the Internet.

From an architectural point of view, the proposed ISM will make an important contribution towards providing security and management of the Internet; it will enable various subnets in the Internet to communicate with one another and to coordinate information in detecting potential attacks. As the Internet becomes larger, this decentralized architecture will avoid an information-flow bottleneck at the central processing node (funnelling point), which would occur under a centralized architecture.

Our future work includes using the NSM as a testbed to analyze various methods of thumbprinting. Not only are we analyzing methods with respect to resolution, efficiency, and semantic content (see Table 1), but we are exploring the possibility of mapping one thumbprint format into a second. For example, a government site may place a greater emphasis on high resolution than a university site, so they would be using two different thumbprint formats. However, if a mapping could be made from the high resolution thumbprint to the low resolution thumbprint, comparisons and tracking could still be performed.

Other future work includes testing, refining, and extending the protocols described here. As we move from design and testing to full implementation, we will probably find flaws in our initial design.

Finally, we are investigating attacks against time protocols, which can in turn subvert the effectiveness of thumbprinting.

References

[Den87] D.E. Denning, "An Intrusion Detection Model," *IEEE Trans. on Software Engineering*, vol. SE-13, no. 2, pp. 222-232, Feb. 1987.

[Den90] P.J. Denning, ed. *Computers Under Attack: Intruders, Worms, and Viruses*. New York: ACM Press, 1990.

[Dow90] C. Dowell and P. Ramstedt, "The COMPUTERWATCH Data Reduction Tool," *Proc. 13th National Computer Security Conference*, pp. 99-108, Washington, D.C., Oct. 1990.

[Heb91] L.T. Heberlein, B. Mukherjee, K.N. Levitt, D. Mansur., "Towards Detecting Intrusions in a Networked Environment," *Proc. 14th Department of Energy Computer Security Group Conference*, May 1991.

[Lun90] T.F. Lunt, et al., "A Real Time Intrusion Detection Expert System (IDES)," Interim Progress Report, Project 6784, SRI International, May 1990.

[Sma88] S.E. Smaha, "Haystack: An Intrusion Detection System," *Proc. IEEE Fourth Aerospace Computer Security Applications Conference*, Orlando, FL, Dec. 1988.

[Sna91] S.R. Snapp, J. Brentano, G.V. Dias, T.L. Goan, L.T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal, D.L. Mansur, "DIDS (Distributed Intrusion Detection System)—Motivation, Architecture, and an Early Prototype," to be published in *Proc. 14th National Computer Security Conference*, Oct. 1991.

[Sto89] C. Stoll, *The Cuckoo's Egg*, Doubleday, 1989.

[Vac89] H.S. Vaccaro and G.E. Liepins, "Detection of Anomalous Computer Session Activity," *Proc, 1990 Symposium on Research in Security and Privacy*, pp. 280-289, Oakland, CA, May 1989.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY/
NATIONAL COMPUTER SECURITY CENTER

15TH NATIONAL COMPUTER SECURITY CONFERENCE

October 13-16, 1992
Baltimore Convention Center
Baltimore, MD